Auto-regressive Graph Generation Modeling with Improved Evaluation Methods

Chia-Cheng Liu[*,†]
University of Toronto
cc.liu2018@gmail.com

Harris Chan[*,†]
University of Toronto
Vector Institute
hchan@cs.toronto.edu

Kevin Luk
Borealis AI
kevin.kh.luk@gmail.com

Abstract

Graphs are frequently used to organize and represent information in various real-world domains such as social sciences, medicine, and language. Unfortunately, generating realistic graphs and evaluating them quantitatively is often notoriously difficult due to the complex structural dependencies inherent to graphs. In this paper, we make two central contributions: first, we propose a novel classifier-based evaluation method to measure graph generation quality and diversity. Secondly, we demonstrate empirically that our attention-based Transformer models achieve competitive performance in terms of generation quality compared to state of the art recurrent models.

1 Introduction

Graph generation is an important problem across a wide range of application domains, such as modeling interactions between physical objects or finding new chemical structures to facilitate drug design. Much of the recent progress in this field have been driven largely by the successes of deep learning. For example, deep generative models have been utilized in both molecular generation [22, 7] and community networks [4][3]. While these approaches work relatively well on small graph datasets, they struggle severely when the size of the graphs increase. An ongoing challenge has been to develop a graph generation method which is both flexible (in domains with complicated dependencies) and scalable (in the number of edges and nodes).

The recent paper of Graph Recurrent Neural Networks (GraphRNN) [27] provides a promising approach towards flexible and scalable graph generation. The main idea behind GraphRNN is to formulate graph generation as a sequential generation problem and use recurrent-based models to generate graphs. Building off of this philosophy, we propose to improve GraphRNN by replacing the recurrent architecture with an attention-based Transformer model, as the latter is better suited to handle long-term dependencies than recurrence. We verify empirically that Transformer's attention mechanism helps to capture the long-term dependencies inherent in sequential graph generation.

Another major issue in graph generation is the lack of meaningful metrics available to quantitatively evaluate the generated graphs. Previous works have mostly relied on comparing the distribution of

[*]Equal contribution
[†]Work done as an intern at Borealis AI
[3]Further related works are discussed in Appendix G.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

graph statistics such as degree sequence, clustering, or number of orbits [2, 24, 27]. Unfortunately, these proposed metrics tend to correspond poorly with human visualization and in addition, they often fail to capture the diversity within the generated graphs.

To this end, we introduce a new classifier-based method using the Graph Isomorphic Network (GIN) from [25]. The intuition of this approach is motivated by the Fréchet Inception Distance [9]; a metric commonly used in the study of image generation, where image quality and diversity are evaluated using the deep features of the Inception Network. In our setting, GIN plays the role of the Inception Network and learns an embedding which captures topological similarities between graphs in the embedding space. As a consequence, metrics derived from GIN embeddings can be expected to correlate better with human perception as compared to graph statistics-based metrics.

Main Contributions. 1) We introduce a novel classifier-based evaluation method based on GIN for graph generation, which is empirically verified to correlate better with human judgement on generation quality and diversity than previous evaluation metrics. 2) We propose a new self attention-based Transformer graph generative model. Under our new evaluation method, the proposed model achieves competitive performance across a range of graph types, and outperforms GraphRNN by a large margin in some cases.

2 Sequential Graph Generation

2.1 Modeling Graph as Sequences

An undirected graph $G = (V, E)$ is specified by its set of vertices V and the set of edges between vertices, $E \subset V \times V$. The graph can be turned into a sequence by choosing an ordering π on the nodes. Denote the ordered vertices by $V^\pi = \{v_1, ..., v_n\}$. The edge information is then represented by a symmetric binary adjacency matrix A^π where $A^\pi_{i,j} = \mathbb{1}[(v_i, v_j) \in E]$. Two graphs are said to be isomorphic to each other if the corresponding adjacency matrices coincide after reordering of the vertices. We write the sequence of adjacency vectors as

$$\text{seq}(G, \pi) = (\tilde{A}^\pi_1, ..., \tilde{A}^\pi_n),$$

where each $\tilde{A}^\pi_i = (\tilde{A}^\pi_{i,1}, ..., \tilde{A}^\pi_{i,i-1}) \in \{0, 1\}^{i-1}$ records the edge relation from the node v_i to its previous nodes.

The key idea of GraphRNN [27] is to generate graphs in a sequential manner. More precisely, an autoregressive model is trained to approximate the distribution $p(\tilde{A}^\pi_i | \tilde{A}^\pi_{i-1}, ..., \tilde{A}^\pi_1)$, so that we can generate graphs by sampling sequentially from $p(\text{seq}(G, \pi)) = \prod_{i=2}^n p(\tilde{A}^\pi_i | \tilde{A}^\pi_{i-1}, ..., \tilde{A}^\pi_1)$.

The architecture of GraphRNN composes of two parts: a graph-state model f_{state} and an output model f_{out}. The graph-state model is an RNN which at each time step i takes as input the vector $\tilde{A}^\pi_i$ and maintains a hidden state h_i recording the state of the graph generated so far. The output model is again an RNN which, at time step i, initiates its hidden state as h_i from the graph-state model, and autoregressively generates one edge $\tilde{A}^\pi_{i+1,j}$ at a time. The entire procedure can be summarized as:

$$h_i = f_{\text{state}}(h_{i-1}, \tilde{A}^\pi_i), \quad \tilde{A}^\pi_{i+1} = f_{\text{out}}(h_i). \tag{1}$$

A simplified version of GraphRNN (termed GraphRNN-S in [27]) generates edges for a given node non-autoregressively. In this case f_{out} is a multi-layer perceptron (MLP) which outputs the Bernoulli parameters for each edge $\tilde{A}^\pi_{i+1,j}$.

2.2 Graph Transformer

The vanilla Transformer in [23] is an encoder-decoder model designed for machine translation tasks. Unlike traditional recurrent architectures employed in these tasks, the Transformer model is fully attention-based, with scaled dot-product attention mechanism.

Our Graph Transformer model, which we denote by **RNN-Transf**, replaces the edge-output RNN model in the GraphRNN architecture with a vanilla Transformer decoder. More precisely, the output model f_{out} contains self-attention sublayers and graph-state attention sublayers with memory from the hidden state of f_{state}. We reformulate (1) as

$$h_i = f_{\text{state}}(h_{i-1}, \tilde{A}^\pi_i), \quad \tilde{A}^\pi_{i+1,j} = \text{Transf}_{\text{out}}(\tilde{A}^\pi_{i+1,<j}, m = (h_1, ..., h_i)), \tag{2}$$

where m denotes the graph-state attention memory. We use the same input positional encoding for Transformers as described in [23]. Further details of the architecture can be found in Appendix B.

3 Evaluation

3.1 Evaluation Based on Graph Statistics

Previous graph generative models have primarily been evaluated using graph statistics. The baseline metric we use for comparative purposes is from [27] where the Maximum Mean Discrepancy (MMD) between generated graphs and the dataset is computed over degree, clustering, and orbit statistics.

However, in Figure 1, we observe inconsistencies between graph visualizations and the numbers evaluated from the MMD metrics. Additionally, graph statistics-based metrics are generally incapable of distinguishing diversity within generated graphs. Hence, a numerical evaluation method which can assess both generation quality and diversity is highly desirable.

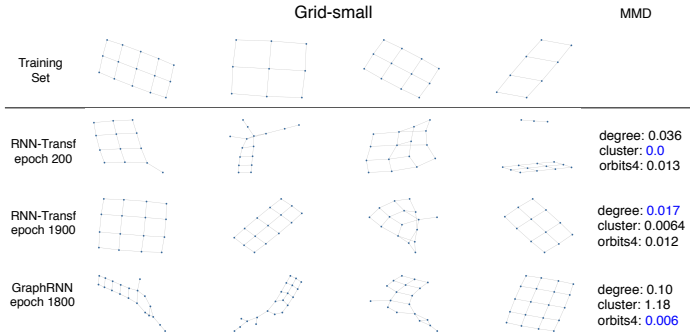

Figure 1: We show sample graphs generated by different generative models trained for different number of epochs on the Grid-small dataset. Examining the visualization shows that RNN-Transf at epoch 1900 generates the most grid-like graphs with more diversity than the other two models. However, MMD metrics using different graph statistics favor different models and hence are inconclusive.

3.2 Evaluation Based on a Graph Classifier

We propose a novel evaluation method which leverages the discriminative power of Graph Neural Networks (GNN) [19]. The expressive power of GNNs were recently studied in [25] and moreover, it was theoretically shown that the maximal expressive power is bounded by the Weisfeiler-Lehman (WL-) graph isomorphism test. As a corollary, the authors in [25] developed the Graph Isomorphism Network (GIN), a prototypical GNN that attains as much discriminative capacity as the WL-test in theory.

We devise our metrics based on the GIN graph classifier to demonstrate the effectiveness of the classifier-based approach. The advantage of using GIN over the traditional WL-subtree kernel method lies in the fact that GIN learns a graph embedding which expresses the similarity among graphs that are non-isomorphic but with similar topological structure. A visualization of the learned graph embeddings using t-SNE is given in Figure 4 in Appendix D.

We propose two GIN-based metrics to measure the quality and diversity of graph generations:

- **Accuracy**: We measure the quality of graph generation conditioned on a given class by the classification accuracy of a GIN pre-trained on a training set consisting of multiple classes.
- **Fréchet Inception Distance (FID)**: FID computes a distance in the embedding space between two multivariate Gaussian distributions fitted to a generated set and a reference set, the latter of which in our case is the entire dataset of the respective class. A lower FID value indicates better generation quality and diversity. In our setting, we estimate the means μ and covariance matrices Σ for the GIN graph embeddings of the generated set X and the testing set $test$, and compute: $\text{FID}(X, test) = ||\mu_X - \mu_{test}||_2^2 + \text{Tr}(\Sigma_X + \Sigma_{test} - 2(\Sigma_X \Sigma_{test})^{1/2})$

4 Experiments

4.1 Training Graph Generative Models

All graph generative models are trained on the *small* graph datasets containing both synthetic and real datasets from [27]: **Community-small**, **Grid-small** and **Ego-small**. Descriptions and visualizations of sample graphs from each datasets can be found in Appendix A. We also use an auxiliary dataset Ladder-small only for training GIN, also from [27].

For GraphRNN and GraphRNN-S baselines, we use the public official implementation available on Github [26]. The **GraphRNN+Attn** is a variant of GraphRNN with attention mechanism in the output level RNN, taking the last M hidden states of the graph level RNN as context for additive attention [3]. **RNN-Transf** is our Graph Transformer model introduced in Section 2.2. We follow the same experimental set up as in [27], and refer the reader to Appendix B for details. We note that the number of parameters in our RNN-Transf model is comparable to that of GraphRNN with the hyperparameters given in Appendix B.

4.2 Evaluation Results

We evaluate our proposed self-attention based models, GraphRNN+Attn and RNN-Transf, with the GIN metric we developed earlier in the paper. Details on the training of GIN can be found in Appendix B.

We plot the value of the GIN and MMD based (comparing the generated graphs against the full dataset) metrics over the course of training in Figure 2 for the Grid-small dataset. For GIN based metrics, we observe generally monotonic behavior improving in the desired direction as training progresses, in contrast to the MMD based metric which appears to be less correlated.

In Table 1, we show train/test accuracies for each trained classifier. We select the models of highest accuracies among all epochs and report the GIN metrics of these models. The GIN evaluation metrics show that the proposed architecture **RNN-Transf** outperforms the baseline GraphRNN by a large margin on the Community-small and Grid-small datasets. Visualizations of the graphs generated by the models reported in Table 1 are shown in Appendix H. For completeness, we report evaluations using the MMD metrics in Appendix F.

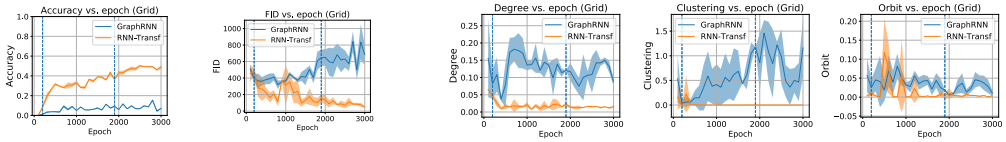

Figure 2: **(1st/2nd)** Our GIN based metrics (Accuracy and Fréchet Inception Distances (FID)) and **(3rd to 5th)** MMD based metrics [27] (Degree, Clustering, Orbit) over training epochs for the GraphRNN and RNN-Transf architecture on the Grid-small dataset. The curve indicates the mean over 3 sets of samples at each epoch and shaded area represents one standard deviation in each direction. See Appendix E for the plots for other datasets. Visualizations of graphs generated by RNN-Transf at epochs 200 and 1900 (indicated by the dotted vertical lines in the plots) are shown in Figure 1.

Table 1: GIN evaluation metrics. Accuracies (Acc.) and Fréchet Inception Distances (FID) are reported on models which achieve best accuracies among all epochs, where FID is computed with reference to entire dataset of respective class. The ↑/↓ symbol indicates that higher/lower is better, respectively.

	Community-small		Ego-small		Grid-small	
Train/Test Acc.	0.98/0.91		0.99/0.97		0.99/0.97	
	Acc. ↑	FID ↓	Acc. ↑	FID ↓	Acc. ↑	FID ↓
GraphRNN (reprod.)	0.36	55.55	**0.60**	**65.16**	0.16	375.24
GraphRNN+Attn (ours)	0.34	79.23	0.20	246.28	0.10	829.12
RNN-Transf (ours)	**0.84**	**16.98**	0.18	255.05	**0.51**	**57.08**

5 Discussion and Future Work

We demonstrated that combining Transformer architecture with recurrent networks for graph generation can be as competitive or even better than a solely recurrence-based approach. We believe that our Graph Transformer model can be further improved by considering a custom relative positional encoding [21, 10], and other sparse attention mechanisms [6]. In addition, we illustrated the shortcomings of existing graph generation evaluation metrics. Our proposed GIN metrics highlights the potential of classifier-based methods in evaluating graph generation. An interesting avenue to pursue in the future is to examine how GIN metrics or other classifier-based evaluation metrics perform on large-scale graph datasets.

Acknowledgments

CL and HC were supported by the MITACS Accelerate Fellowship.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] David L Alderson and Lun Li. Diversity of graphs with highly variable connectivity. *Physical Review E*, 75(4):046102, 2007.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.

[5] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[6] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv e-prints*, April 2019.

[7] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[8] P Erdős and A Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[10] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. 2018.

[11] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2328–2337, 2018.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL `https://doi.org/10.18653/v1/P17-4012`.

[14] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11(Feb):985–1042, 2010.

[15] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[16] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/parmar18a.html`.

[17] Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

[18] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social networks*, 29(2):173–191, 2007.

[19] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[21] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[22] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[24] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[25] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

[26] Jiaxuan You. Graphrnn: Generating realistic graphs with deep auto-regressive model. `https://github.com/snap-stanford/GraphRNN`, 2018.

[27] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5694–5703, 2018.

A Datasets

We train our graph generative models on the following datasets. Their descriptions are given here and sample visualizations can be found in Appendix H.

Community-small. 100 two-community graphs with $12 \leq |V| \leq 20$, $M = 20$. Each community is generated by the Erdős-Rényi model (E-R) [8] with $n = |V|/2$ nodes and $p = 0.8$. We then add $0.05|V|$ inter-community edges with uniform probability.

Grid-small. 12 standard 2D grid graphs with $4 \leq |V| \leq 20$, $M = 15$, ranging from 2×2 to 4×5 arrangements.

Ego-small. 200 1-hop ego networks extracted from the Citeseer network [20] with $4 \leq |V| \leq 18$, $M = 15$. Nodes represent documents and edges represent citation relationships.

B Architecture and Experiment Details

B.1 Graph generative models

The Gated Recurrent Unit architecture is chosen for all RNNs in the experiments, following the official open sourced implementation of GraphRNN [26]. Our Transformer implementation is adapted from OpenNMT [13].

A layer in our Transformer edge output model consists of a multi-head self-attention sublayer and a multi-head inter-attention sublayer with context from the hidden state of the graph state RNN, followed by a fully-connected feed-forward sublayer, with residual connection and layer normalization applied to the output of each of the sublayers.

Following the notation in Section 2.1, we illustrate the generation process of our Graph Transformer model (RNN-Transf) at inference time in Figure 3.

To cut down on complexity, Breadth First Search (BFS) is used to determine the node ordering of the graph. Doing so reduces the length of $\tilde{A}_i^\pi$ when i is large, to a maximum length $M \leq n$ for many cases. Figure 3 illustrates the maximum length $M = 2$ for the corresponding graph when using BFS ordering. Moreover, BFS reduces the number of possible ways to order isomorphic graphs.

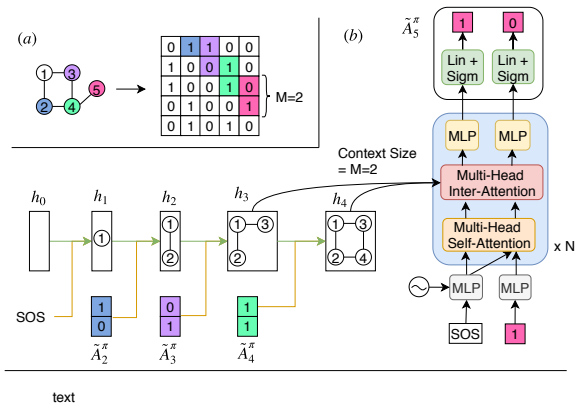

Figure 3: (a) An example graph and its corresponding adjacency matrix. Highlighted are the relevant parts of each column as the adjacency vector $\tilde{A}_i^\pi$ (reversed order). Due to BFS node ordering scheme, we can limit the dimension of $\tilde{A}_i^\pi$ to $M = 2$. (b) One of our proposed model, **RNN-Transformer** at inference time. The first four nodes have been generated. We use an RNN to encode the graph state in its hidden vector h_i. An edge-output Transformer uses the last M RNN hidden states as memory context to predict the adjacency vector $\tilde{A}_5^\pi$ for node 5 (Sec 2.2).

We use 80% of the graphs in each dataset for training and the remainder for test. The objective is to maximize the log-likelihood of the training dataset. We train for a fixed budget of 3000 epochs with each 32 batches in each epoch and batch size 32, using the Adam optimizer [12].

In all our experiments involving Transformer models we use the vanilla Transformer with the following modifications and specifications:

- We pass the input through an MLP (instead of an input linear embedding) before feeding into the Transformer. This is the same practice employed in GraphRNN.
- We set the dimension of the feed-forward fully-connected sublayer in Transformer equal to the hidden unit size.

Hyperparameters	Transformer	RNN
Hidden units dim	16	64(graph-state); 16(output)
Input MLP dim	8	32(graph-state); 8(output)
Number of layers	2	4(graph-state); 4(output)
Number of heads	4	-
Dropout rate	0.1	-
Optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$)	Adam($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)
Learning rate	Scheduled according to formula (3)	0.003, decay by 0.3 at epoch 400,1000

Table 2: Hyperparameters for graph generative models

- Dropout with dropout rate 0.1 is applied to each sublayer output and the sum of positional encoding and input, as in vanilla Transformer.
- We adopt the same learning rate schedule as in vanilla Transformer:

$$lr = d^{-0.5} \cdot \min(step^{-0.5}, step^{-0.5} \cdot 400^{-1.5}), \tag{3}$$

  where d is the hidden unit size and $step$ is the step number.

For the GraphRNN+Attn variant, we implement an additive attention mechanism in the output-level RNN, taking the last M hidden states of the graph level RNN as context vectors $h_{n-M,n}^{graph-state} \in \mathbb{R}^{d \times M}$ for n generated nodes. If $n < M$, then we only take the last n timesteps. We still initialize the initial hidden state of the output-level RNN with the current hidden state of the graph-level RNN hidden state. We compute the attention weights at time t (for the t-th edge output by the output RNN) $\alpha^{(t)} \in \mathbb{R}^M$ by concatenating the flattened context vectors and the current output RNN hidden state $h_t^{output} \in \mathbb{R}^{d \times 1}$ and pass through a single layer MLP with $d = 16$ hidden units:

$$\tilde{\alpha}_i^{(t)} = W_2(\max(0, W_1[h_t^{output}; h_{n-M,n}^{graph-state}] + b_1)) + b_2 \tag{4}$$

$$\alpha_i^{(t)} = \text{softmax}(\tilde{\alpha}^{(t)})_i, \tag{5}$$

$$c_t = \sum_{i=1}^{M} \alpha_i^{(t)} h_{n-M,n}^{graph-state} \tag{6}$$

Note that while the hidden state for the graph-state RNN is typically larger than the hidden state for the output RNN, we first apply a linear projection to the former so that the dimension matches.

Other hyperparameter configurations are shown in Table 2.

B.2 GIN

Hyperparameters	GIN
Hidden units dim	64
Number of layers	5
Number of MLP layers	2
Final layer dropout rate	0.5
Optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)
Learning rate	0.01 with decay rate 0.5 and step size 50

Table 3: GIN hyperparameters

We use the vanilla 5-layer GIN, where in each layer we use a 2-layer MLP of hidden dimension 64. We set the node features as one-hot vectors representing degrees of the respective nodes. The GIN outputs graph embedding vectors and we pass the embedding vectors through a linear embedding, followed by a final layer Dropout with rate 0.5 to get the logits for classification. We train GIN for 60 epochs on 4 classes of small graphs: Grid-small, Ladder-small, Ego-small, and Community-small,

plus a class of 'bad' graphs sampled from generated graphs by GraphRNN-S at epoch 100. We use 90% of the graphs in each dataset for training and the remainder for test, following [25]. Other hyperparameter configurations are shown in Table 3.

C Architecture Computational Complexity Comparison

As in the GraphRNN paper [27], we use teacher forcing during training. Since Transformer models are non-sequential in nature (avoiding recurrences), they are much more parallelizable than RNNs when training on longer sequences. However, at inference time, using a Transformer requires recomputing attentions for all previous positions in a sequence at each time step, thus resulting in higher time computational complexity than using an RNN.

Table 4 summarizes the inference time computational complexity of various architecture variants when generating a graph from start to finish. We denote n to be the number of nodes, M the maximum number of previous nodes to be connected to after BFS ordering, k the number of hidden units in the layer, and d the number of layers.

The first term refers to the complexity for the graph-state model. The $O(Mk)$ term is for embedding the binary adjacency vector to the hidden dimension. The RNN is linear in number of nodes (if we ignore the $O(Mk)$ factor).

The second term refers to the overall complexity for the edge-output model. The factor of $O(n)$ is for repeating the forward pass of the edge-output n times. We see that Transformer model is in between the RNN approach and Additive Attention RNN in terms of complexity, while MLP is the cheapest.

Model Name	Graph	Edge	Inference
GraphRNN-S	RNN	MLP	$O(n(Mk + k^2d)) + O(nk^2d)$
GraphRNN	RNN	RNN	$O(n(Mk + k^2d)) + O(nMk^2d)$
GraphRNN+Attn	RNN	AttnRNN	$O(n(Mk + k^2d)) + O(nM^2k^2d)$
RNN-Transf	RNN	Transf	$O(n(Mk + k^2d)) + O(n(M^2k + Mk^2)d)$

Table 4: Computational complexity comparison between different architecture variants

D tSNE Visualization of GIN graph embeddings

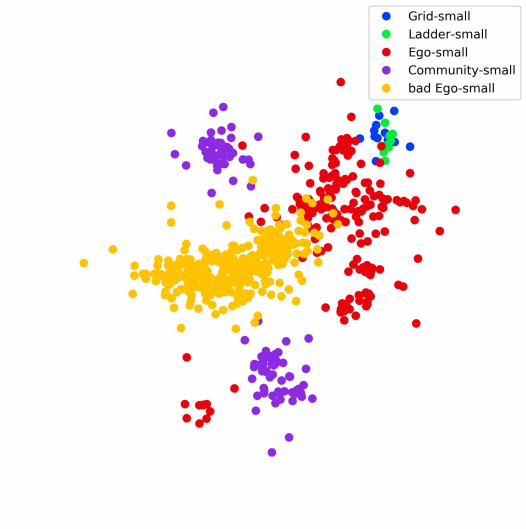

Figure 4: t-SNE visualization of GIN graph embeddings used to evaluate generated Ego-small graphs. GIN is trained to epoch 60 with datasets specified in the figure legend. Graphs of similar structure are clustered in the embedding space. Notably, similarities between grid and ladder graphs, as well as some overlaps of real and 'bad' Ego-small graphs, are observed.

E Comparison of GIN and MMD Metrics

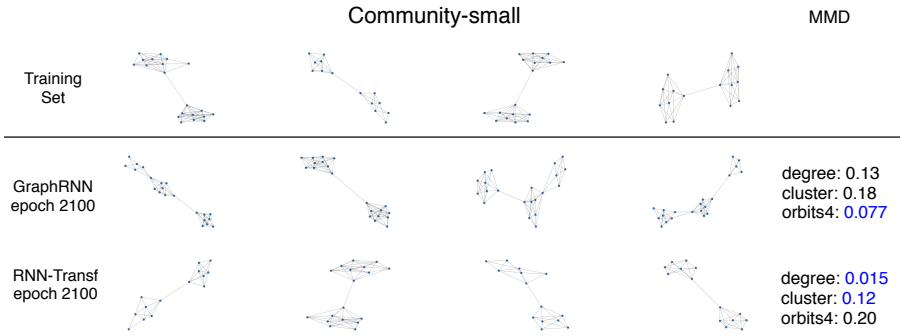

Figure 5

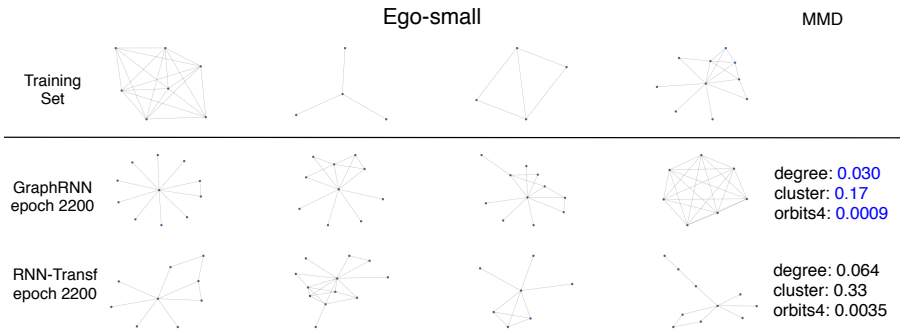

Figure 6

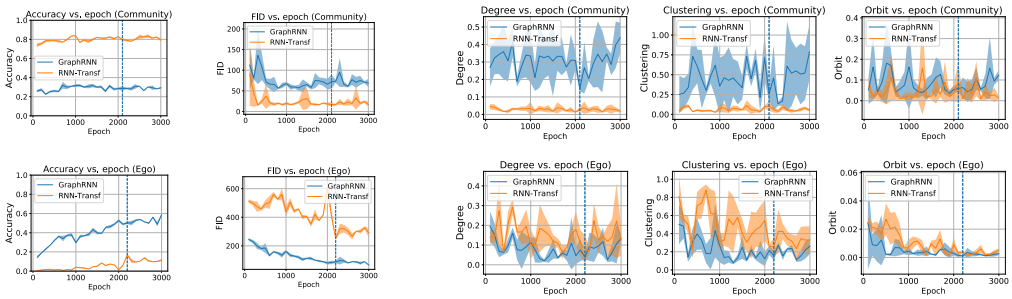

Figure 7: **(1st/2nd column)** Our GIN based metrics (Accuracy and Fréchet Inception Distances (FID)) and **(3rd to 5th column)** MMD based metrics [27] (Degree, Clustering, Orbit) over training epochs for the GraphRNN and RNN-Transf architecture on the Community-small **(top)** and Ego-small dataset **(bottom)**. Visualizations of graphs generated by RNN-Transf at epochs 2100 (Community-small) and 2200 (Ego-small) for (indicated by the dotted vertical lines in the plots) are shown in Figure 5 and Figure 6, respectively.

We further illustrate the shortcoming of the statistics-based MMD metrics used in You et al. [27] on the Community-small dataset. As shown in Figure 5, the Community-small graphs generated by GraphRNN trained to epoch 2100 contains more (3) communities than specified (2) in the training and testing graphs. On the contrary, most graphs generated by RNN-Transf trained to epoch 2100 faithfully represent graphs of 2 communities. However, the MMD orbits metric strongly favors the generation of poorer quality, leading to an overall inconclusive evaluation.

Then we assess the MMD metrics on Ego-small. From visualizations in Figure 6, GraphRNN is able to generate a variety of graphs shapes close to the testing graphs in structure, whereas RNN-Transf can only generate a specific type (the star-shaped) of graph with some glitch. In this case MMD

metrics are able to distinguish the generation quality difference, but it is not clear whether MMD evaluations measure the diversity within the generated samples.

In Figure 7 we compare the MMD metrics with our proposed GIN metrics (Accuracy and FID) on Community-small and Ego-small, all measured every 100 epochs from epoch 100 up to epoch 3000. Again, the MMD metrics generally have higher variance than GIN metrics and appear to be less correlated with human perception of generation quality on Ego-small, whereas our GIN metrics are more in line with human judgement.

F Additional Tables

Table 5 reports the MMD evaluation metrics of the epoch on the entire dataset (train and test), for baselines and our proposed self-attention based variants for the three datasets. For each MMD metric, we find the epoch with the lowest MMD value for that metric on the validation set, then report the corresponding full dataset MMD performance (i.e. early stopping). Table 7 reports the MMD evaluation metrics on the test set, as done by You et al. [27]. The conclusion on the best models based on MMD are different depending on which sets of data are used to report the final performance. Using the test set MMD value corresponds to measuring generalization to out of training examples, while using the entire dataset corresponds to matching the distribution in general.

Table 5: MMD metrics on the full dataset (train and test) of models chosen according to best validation MMD metric value among epochs.

	Community-small			Ego-small			Grid-small		
	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit
GraphRNN (reprod.)	0.148	0.134	0.042	**0.025**	**0.079**	0.004	0.075	0.036	0.018
GraphRNN+Attn (ours)	0.229	0.225	0.020	0.080	0.411	0.003	0.062	**0**	0.012
RNN-Transf (ours)	**0.017**	**0.047**	**0.014**	0.065	0.237	**0.001**	**0.015**	**0**	**0.004**

Table 6: MMD metrics of models chosen according to best GIN accuracies among epochs. The models selected agree with those selected in Table 1.

	Community-small			Ego-small			Grid-small		
	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit
GraphRNN (reprod.)	0.18	0.26	**0.034**	0.16	0.31	0.0034	0.15	0.36	0.034
GraphRNN+Attn (ours)	0.64	0.19	0.60	0.053	0.50	0.0035	0.088	**0**	0.065
RNN-Transf (ours)	**0.030**	**0.077**	0.17	**0.019**	**0.14**	**0.0021**	**0.014**	**0**	**0.0044**

Table 7: MMD metrics for published and reproduced results for GraphRNN, compared to our proposed architectures. Note: there was no published results for Grid-small dataset.

	Community-small			Ego-small			Grid-small		
	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit	Deg.	Clust.	Orbit
GraphRNN [27]	0.03	0.03	0.01	0.0003	0.05	0.0009	-	-	-
GraphRNN (reprod.)	0.080	0.088	0.010	0.063	0.214	0.004	**0.328**	**0**	0.103
GraphRNN+Attn (ours)	**0.016**	**0.011**	**0.002**	0.051	0.256	**0.001**	**0.328**	**0**	**0.093**
RNN-Transf (ours)	0.020	0.057	0.004	**0.014**	**0.051**	0.006	**0.328**	**0**	0.103

G Additional Related Works

Graph Generation. Generative modeling for graphs have a long history and are well-studied. Notable early work in this area include Barabási-Albert model [1], small-world model [24], Kronecker graphs [14], and Exponential Random Graph Models [18]. Such models are rather restrictive in their

scope: they are hand-designed to learn very specific families of graphs and typically incapable of modeling real-world examples.

Recent advances in deep learning have led to a number of neural network based approaches towards graph generation. Graph neural networks were used in [15] to learn probabilistic dependencies of the nodes and edges in a graph. GraphVAE [22] applied variational autoencoders (VAEs) towards the molecule generation task while MolGAN [7] took an implicit, likelihood-free approach by using Generative Adversarial Networks (GANs) for the same problem.

Transformer Models. While Transformer architectures have largely been used for neural machine translation, there have been a number of applications on generation tasks. Huang et al. [10] modified the vanilla Transformer with relative attention mechanism to generate musical compositions with long-term structure. For image generation tasks, Parmar et al. [16] restricted the self-attention mechanism in Transformers to attend to local neighborhoods, achieving state-of-the-art performance on ImageNet. Our work is similar in the sense that we found a use case external to natural language processing and one where applying Transformer yielded considerable improvements.

Evaluation Metrics. Finding an appropriate metric to quantitatively evaluate generated graphs is a notoriously challenging problem. Previous proposals have mostly focused on using graph statistics. [14] made comparisons based on degree distribution and diameter of a graph. Building on this work, You et al. [27] evaluated the MMD over degrees, clustering coefficients, and orbit counts for a wide range of graph datasets. In a similar spirit, Bojchevski et al. [4] utilized link prediction performance to evaluate the generalization power of their proposed model. The common drawbacks to statistics-based metrics are that they often have difficulties capturing generation diversity and tend to correlate poorly with human perception. On the contrary, our classifier-based method using GIN accurately captures diversity and agrees with human perception in many settings as shown earlier.

In molecular generation, evaluation measures based on validity, novelty, and uniqueness have been used in [7]. Brown et al. [5] conducted a more systematic study on benchmarking generative models for *de novo* molecular design using distribution-learning and goal-directed benchmarks. Among the metrics considered in [5], the Fréchet ChemNet Distance [17] also takes inspiration from FID and utilizes ChemNet, which takes the SMILES string representation of a molecular graph and passes it into a neural network composed of 1D convolution, max-pooling, and LSTM layers. Unfortunately, the SMILES representation is not unique [11], and the ChemNet architecture does not remedy the non-uniqueness problem. On the other hand, the GIN network that we use is completely invariant under graph isomorphisms; which is in fact one of the defining components of the GIN architecture [25]. Additionally, these evaluation methods are very domain-specific; as they depend on node and edge features in a molecular graph, or a specific form of representation, e.g. SMILES. Our method in this paper is general purpose and can be applied towards any type of graph.

H Additional Graph Visualizations

We visualize the graphs generated by each of our models on the three datasets in Figure 8. The graphs presented here are cherry-picked to show that all model variants (except GraphRNN-S) can generate at least one graph of decent quality. However, on generation of grid graphs we observed that our **RNN-Transf** variant was able to generate a variety of width/height combinations, while the GraphRNN baseline only generates 4×5 grid graph when the generated graph is indeed of grid shape. The generation diversity improvement in our RNN-Transf is further confirmed by our proposed GIN evaluation method in Section 4.2.

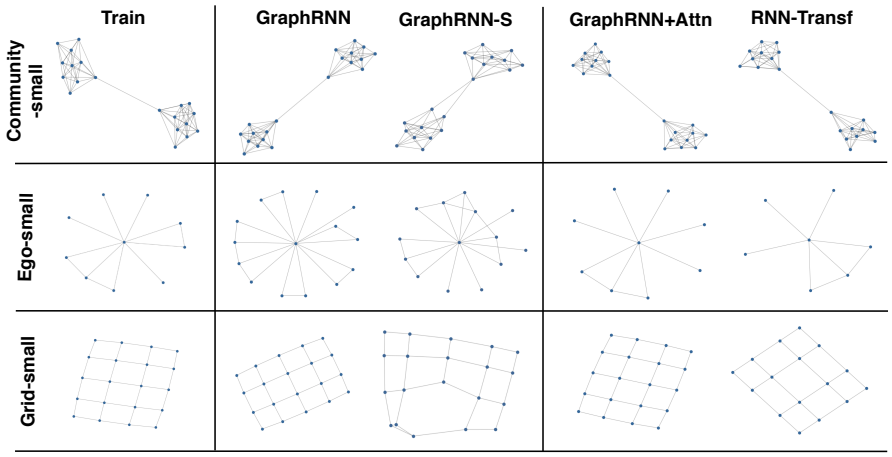

Figure 8: Visualization of graphs for Community-small (top), Ego-small (middle), and Grid-small (bottom) dataset for several architecture variants. The first column shows training set graph, second column group shows baseline GraphRNN and GraphRNN-S, and the third column group contains our proposed self attention-based variants. We use the spring visualization layout for all graph datasets.